IBM Software Group

Steps to Better Requirements Management

Business Analyst World 2010

John MacLeod - <john.macleod@au1.ibm.com>

Agenda

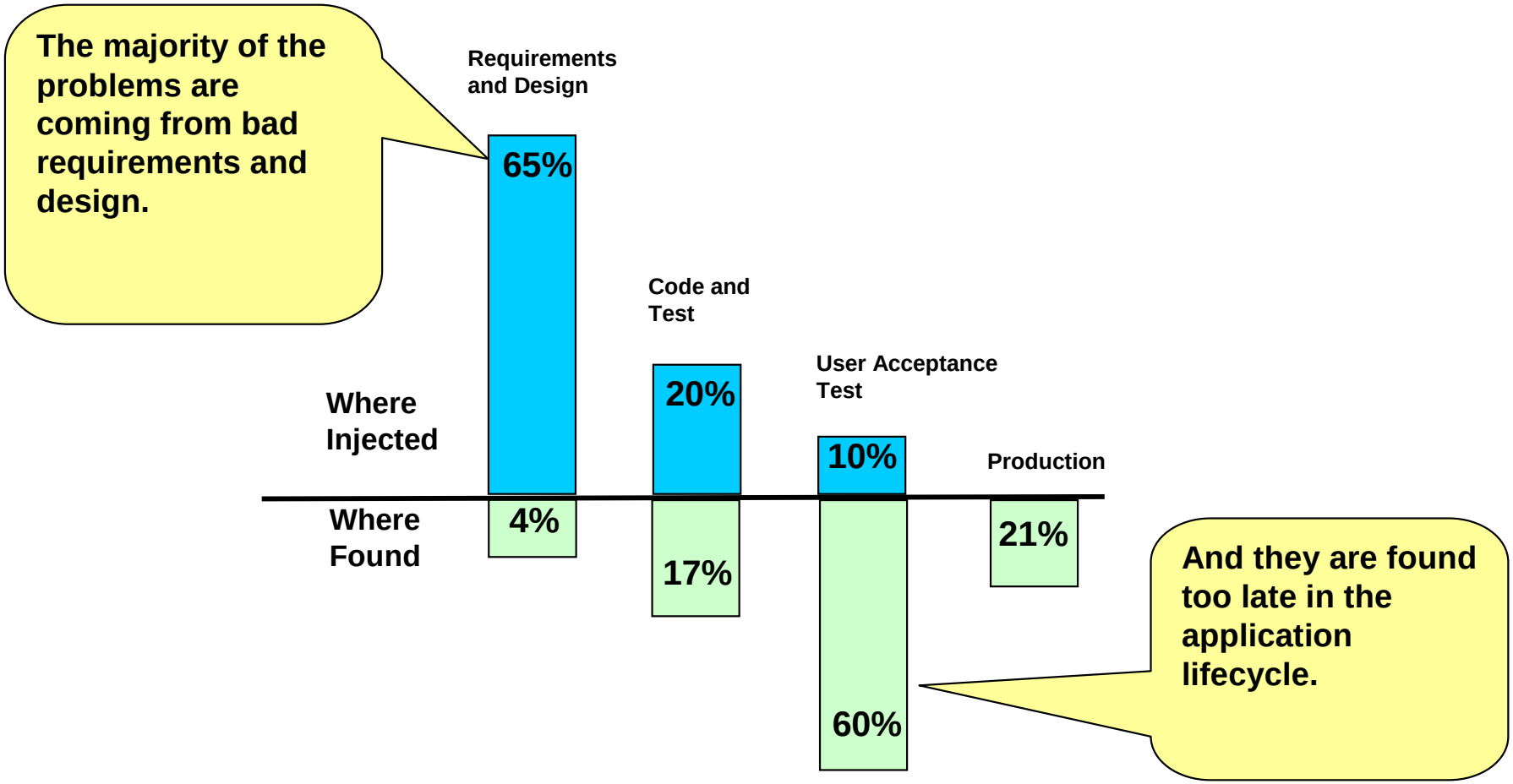
- The Case for Improving Requirements Management
- Good Requirements Management Practices
- No Excuses!

Requirements Management - The Driver

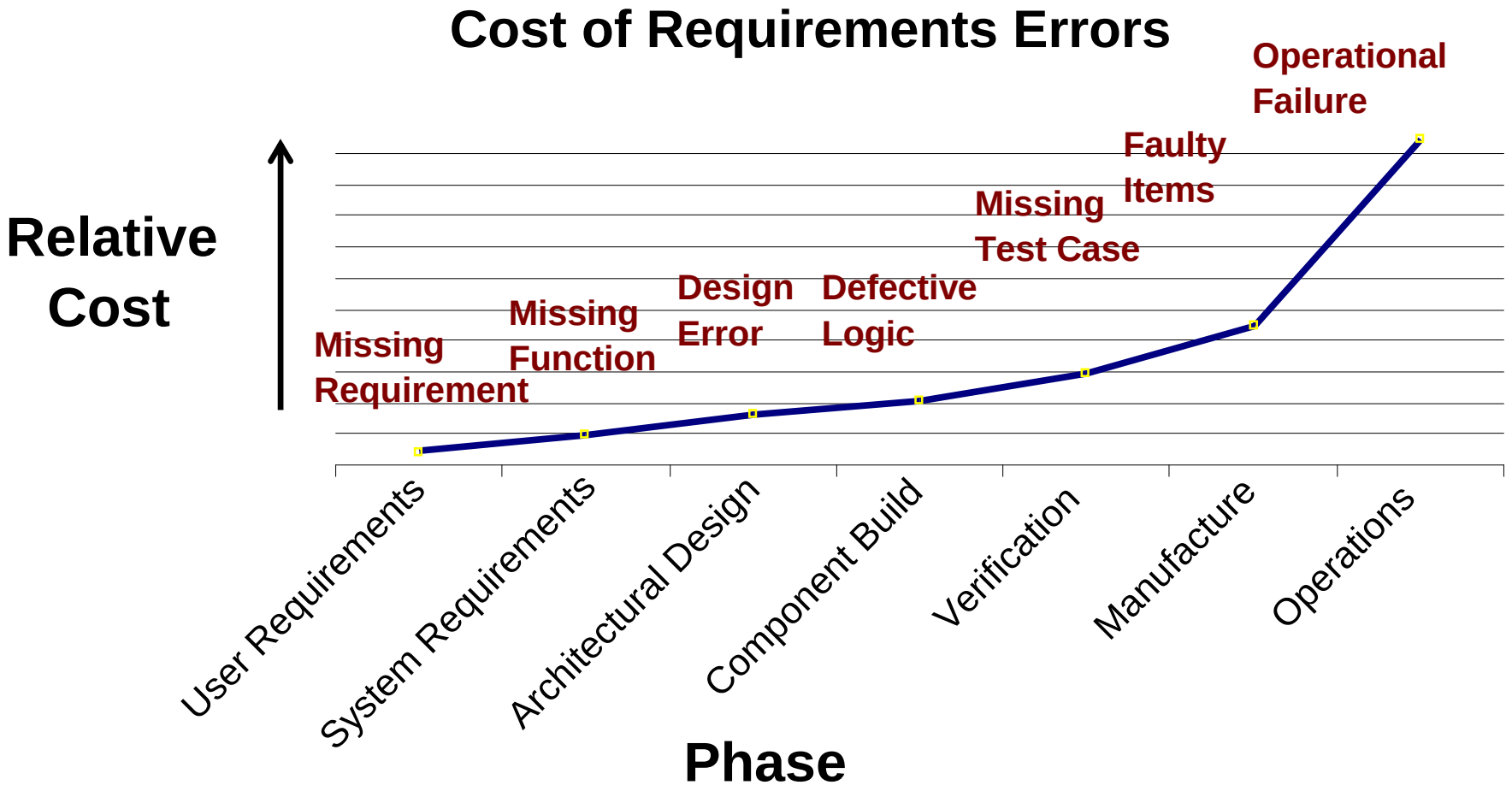
"Analysts report that as many as 71 percent of software projects that fail do so because of poor requirements management, making it the single biggest reason for project failure"

CIO Magazine

Start at the beginning.



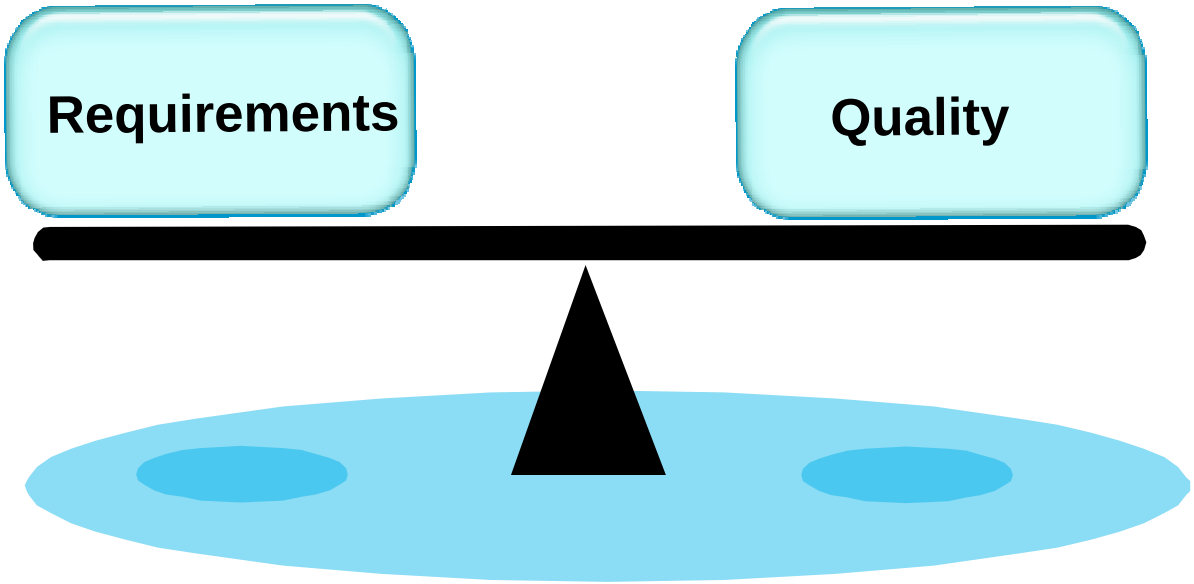
Cost of requirements errors



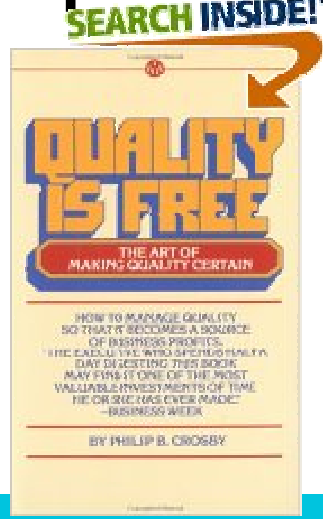
The message is simple. The later you catch an error the more it costs to fix!
It will never be cheaper to catch errors than in the Requirements Phases.

Why are better requirements needed?

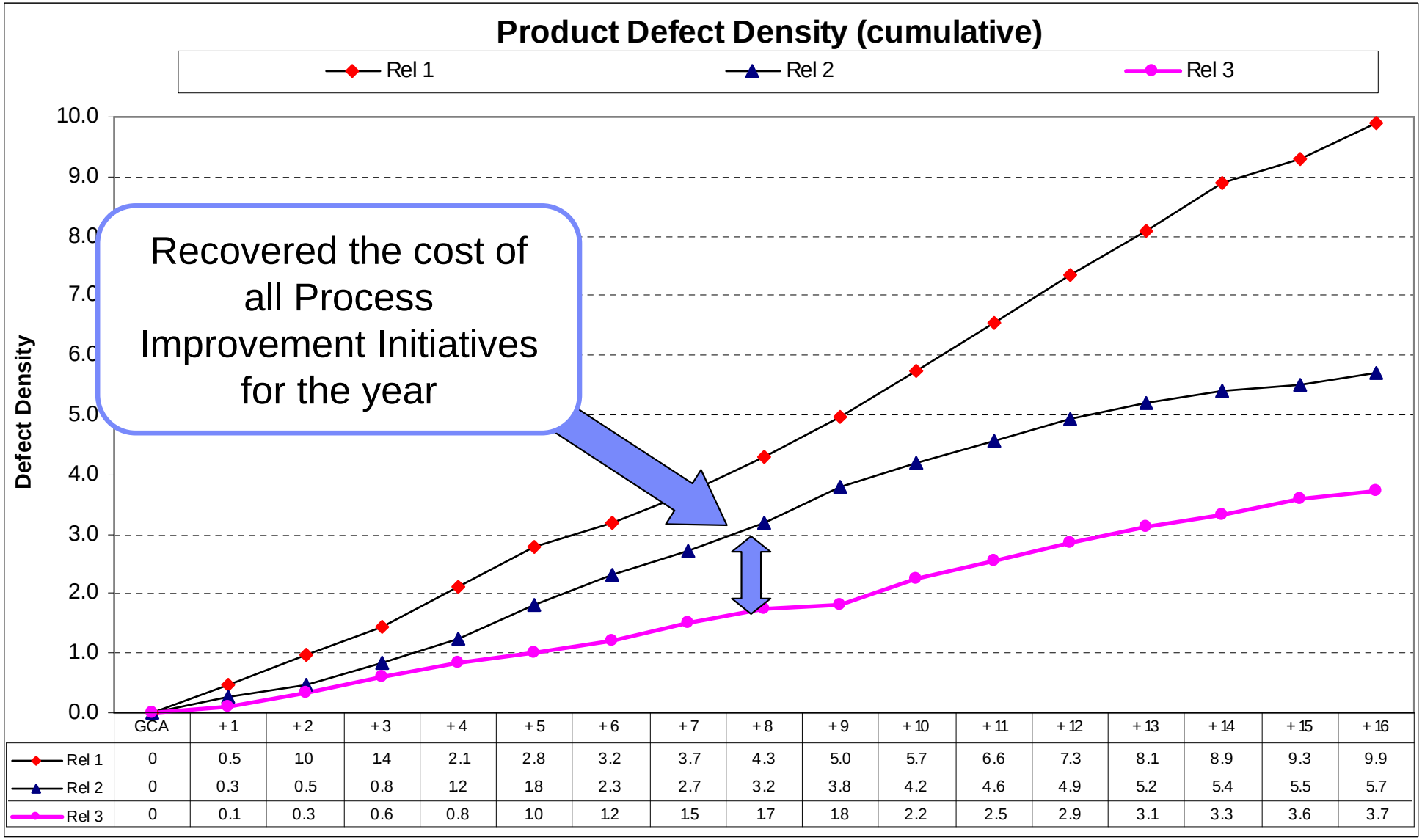
Requirements Management is a High Leverage Activity



“Quality is free”
Phillip Crosby

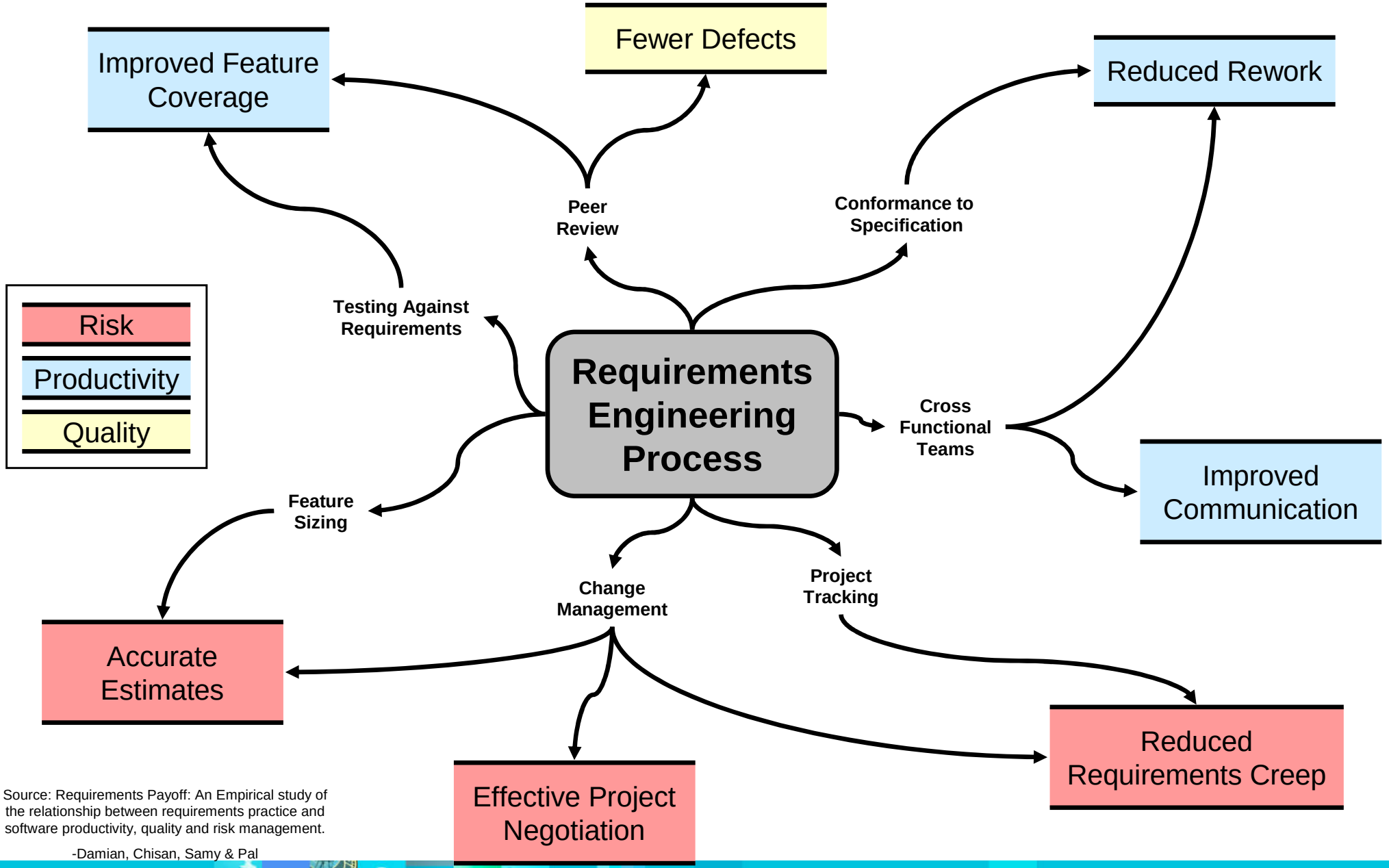


Unisys - Return on Investment



	GCA	+1	+2	+3	+4	+5	+6	+7	+8	+9	+10	+11	+12	+13	+14	+15	+16
Rel 1	0	0.5	10	14	2.1	2.8	3.2	3.7	4.3	5.0	5.7	6.6	7.3	8.1	8.9	9.3	9.9
Rel 2	0	0.3	0.5	0.8	12	18	2.3	2.7	3.2	3.8	4.2	4.6	4.9	5.2	5.4	5.5	5.7
Rel 3	0	0.1	0.3	0.6	0.8	10	12	15	17	18	2.2	2.5	2.9	3.1	3.3	3.6	3.7

Impact of Requirements Practice - Unisys



Source: Requirements Payoff: An Empirical study of the relationship between requirements practice and software productivity, quality and risk management.

-Damian, Chisan, Samy & Pal

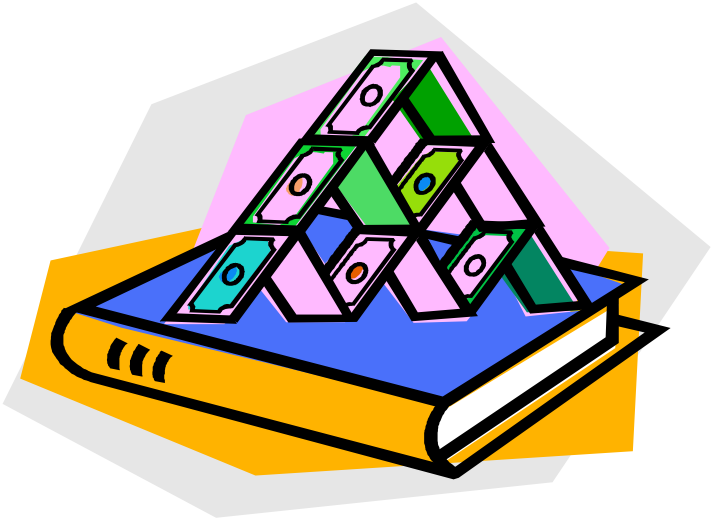
Requirements Are Everywhere

Requirements

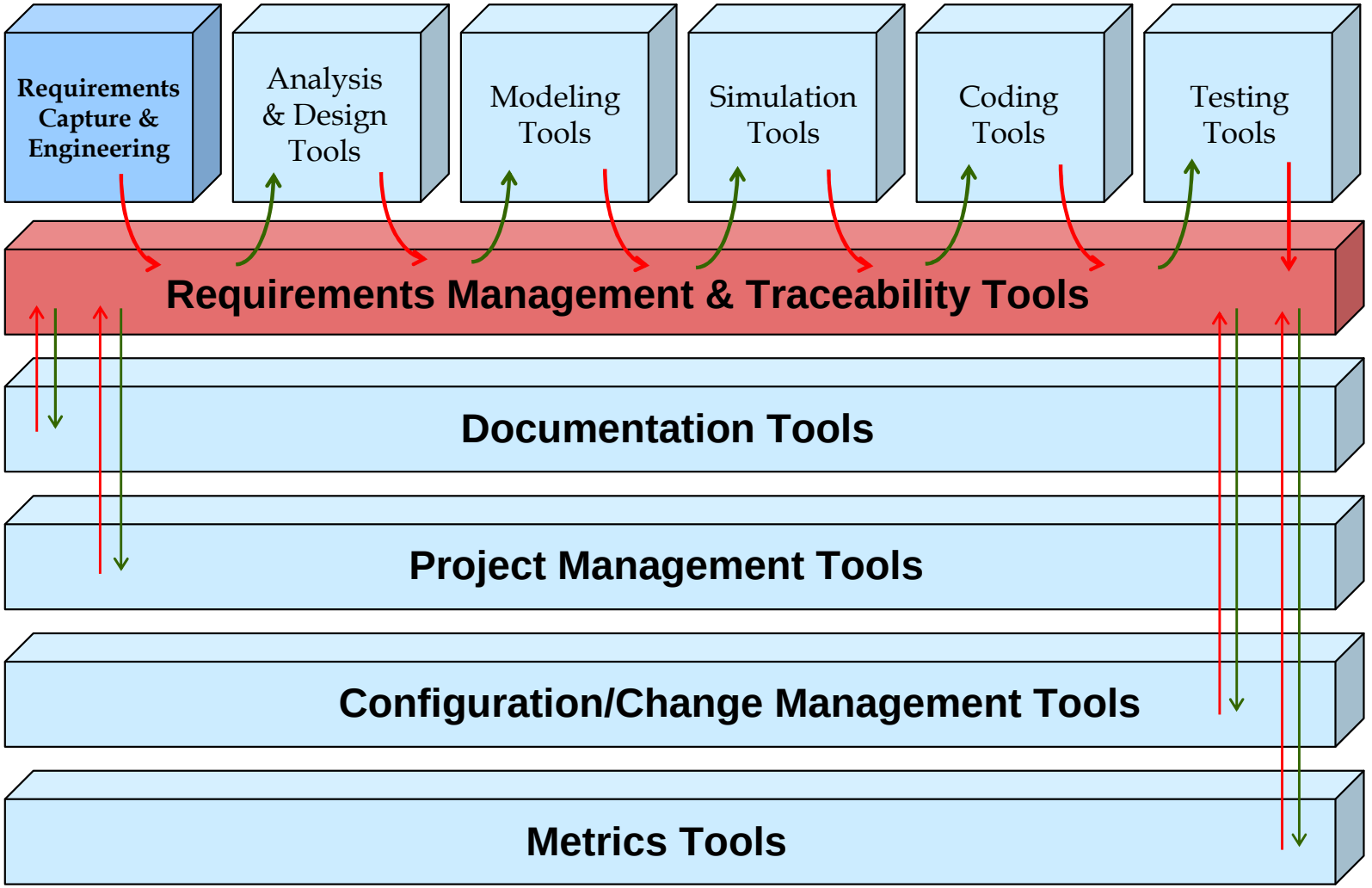
Whole-life Management

Requirements form the basis for:

- project planning
- risk management
- acquisition management
- trade-off
- change control
- qualification / testing
- deployment
- maintenance / support / enhancements
- retirement / disposal



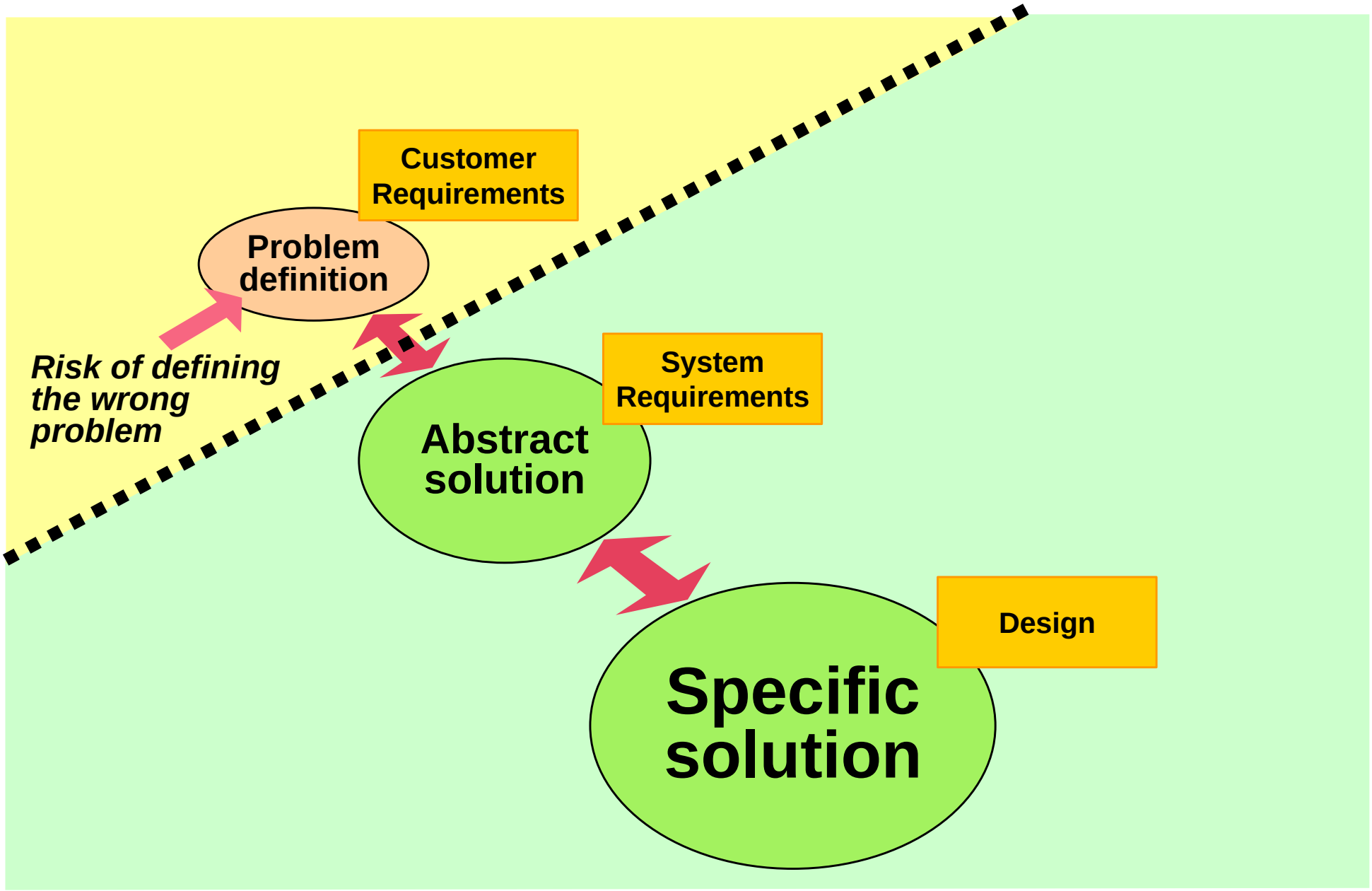
Requirements through the lifecycle



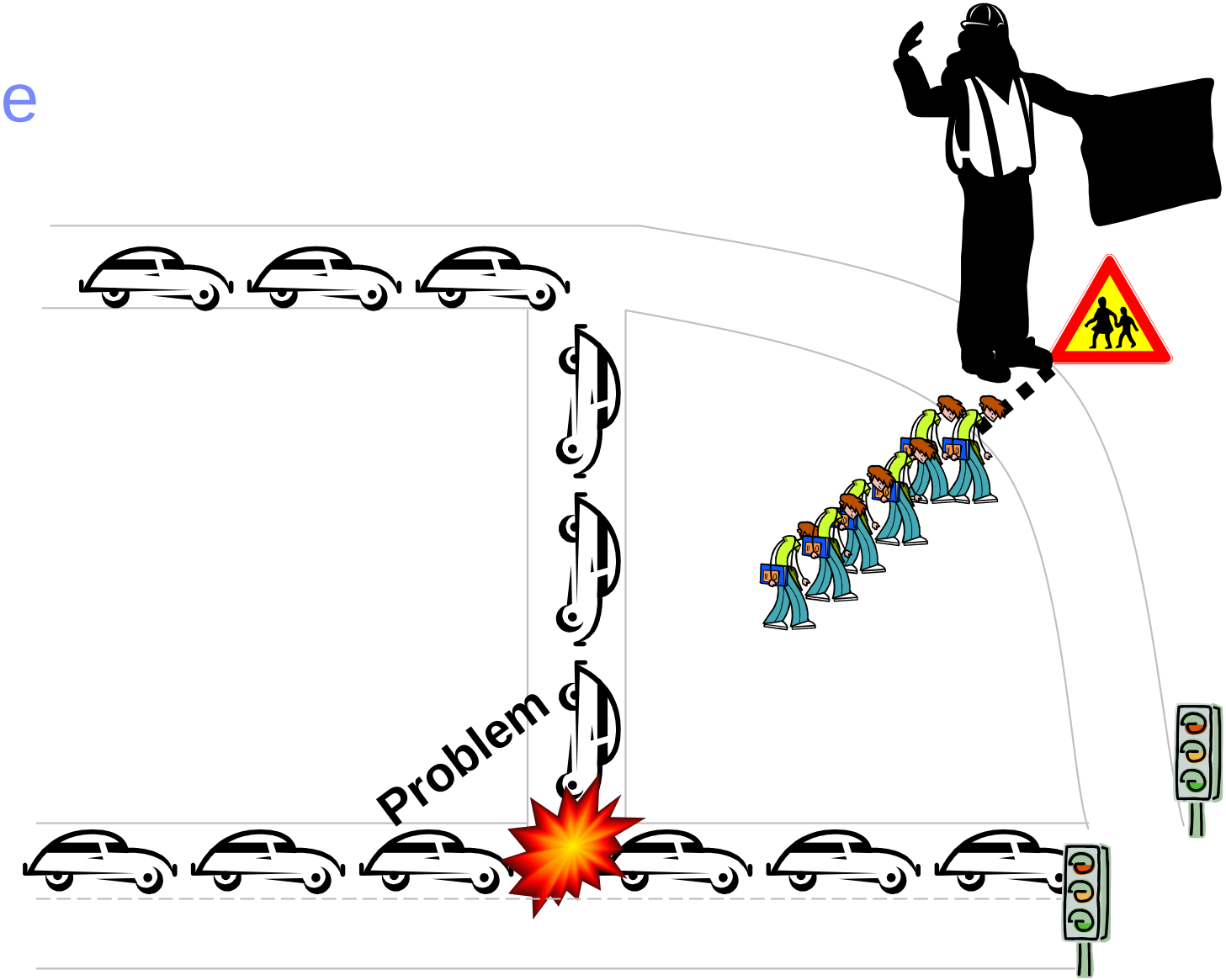
GOOD PRACTICES

0. Recognise you may be up for more work, not less!
1. Understand the difference between the problem & the solution
2. Use concise, clear, consistent language in statements
3. Focus on documents as well statements
4. Drive testing from requirements
5. Create, review and use traceability

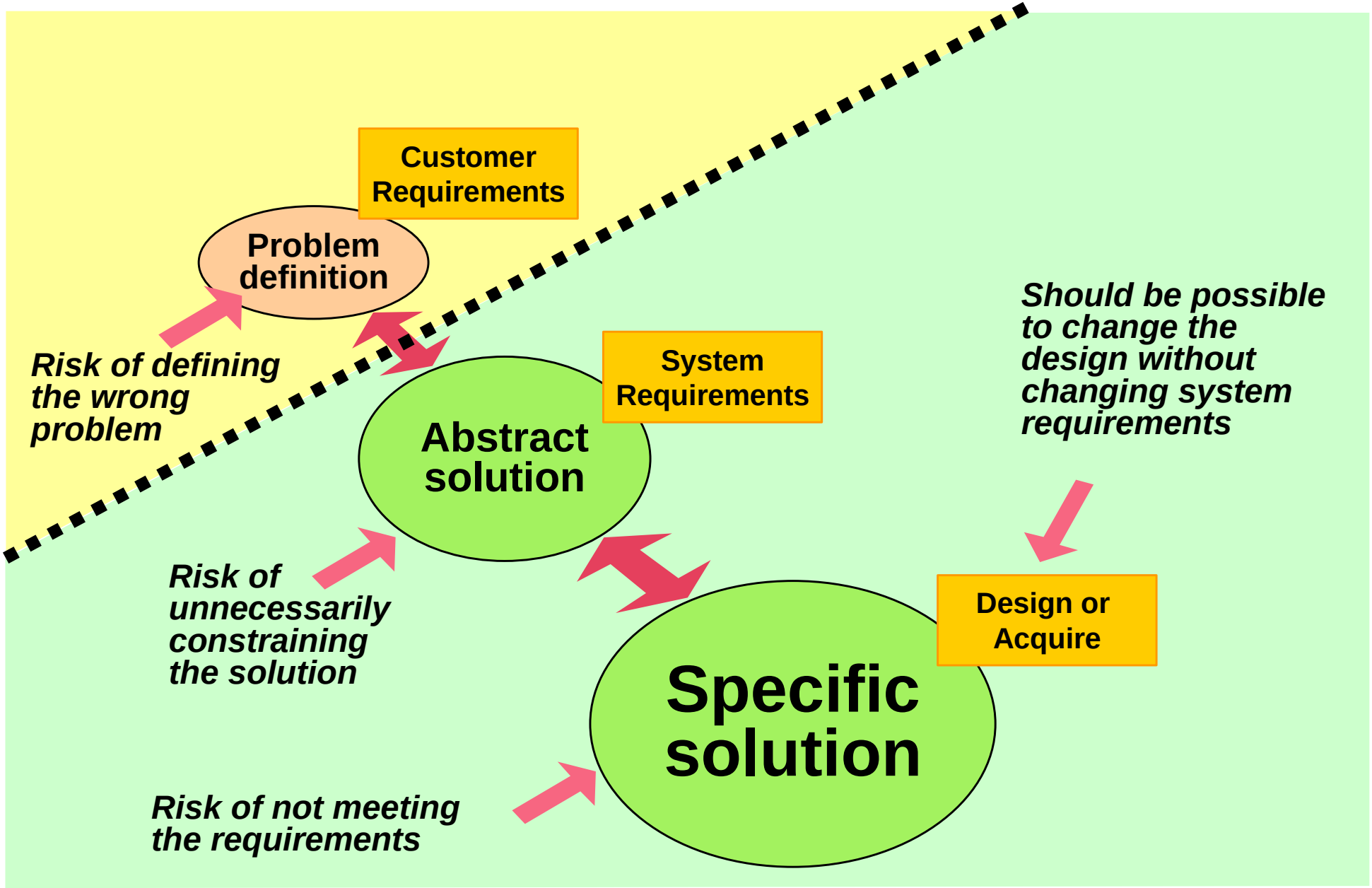
Good Practice 1: Distinguish between Problem and Solution



Example



Good Practice 1: Distinguish between Problem and Solution



Differentiating Problem and Solution

Problem	***Solution***
Customer requirements	***System requirements***
• A description of the problem and its context	• An *abstract* representation of the solution
• Results that stakeholders want from the system	• What the system does
• Do not define the solution, other than for environment	• Do not define the design
• Quality of results	• How well it does it
• Owned by stakeholders or their representatives (e.g. marketing)	• Owned by systems engineers

"The user shall be able to ...." ***"The system shall do …."***

Good Practice 2: Use concise, clear, consistent language

Each requirement statement should be:

1. Individual: each statement is a single traceable element
2. Unique: each statement is uniquely identified
3. Clear: each statement is clearly understandable
4. Precise: each statement is precise and concise
5. Abstract: does not impose a solution on the next layer
6. Testable: each statement can be validated/verified
7. Quantified: each statement has acceptance criteria

Six Things to Avoid

1. Rambling: conciseness is a virtue
2. Let-out clauses: such as “if that should be necessary”; they render the requirements useless
3. Multiple requirements: often indicated by “and”, “or”, “but”, “however”
4. Vague terms: usually, generally, often, normally, typically, user friendly, versatile, flexible
5. Wishful thinking: “100% reliable”, “please all users”, “run on all platforms”, handle all unexpected failures”, “upgradeable to all future situations”
6. Speculation: stick to what you know

Good Practice 3: Focus on documents as well as statements

- Need to balance two aspects:

- Making each requirement statement manageable
  - Focus on the individual statement of requirement (…later)

- Making the requirements document understandable
  - Focus on the requirements document structure

Specifications Contain Statements

Two concerns:

- Focus on the individual statement of requirement:
  - Language
  - Clarity, preciseness
  - Identity, traceability

Statements provide precision

- Focus on the requirements document:
  - Understanding context
  - Assessing completeness
  - Identifying repetition/conflict
  - Navigating/searching requirements

Documents provide context

Seven Criteria for Requirements Documents

Each requirements set should be:

1. Complete / Sufficient: all requirements are present
2. Consistent: no two requirements are in conflict
3. Non-redundant: each requirement is expressed once
4. Modular: requirements statements that belong together are close to one another
5. Structured: there is a clear structure to the requirements document
6. Satisfied: the appropriate degree of design traceability has been achieved
7. Evaluated: the appropriate degree of test traceability has been achieved

Define an outline structure at the outset, and improve it as you go

Good Practice 4: Drive Testing from Requirements

- Of every requirement statement, ask:
  - “How will you know if the need has been met?”
- Improves the way the requirement is expressed
  - Is it quantified?
  - What are the success criteria?
  - Add requirements to make system testable
- Plan the tests now, not later:
  - What kind of tests will be used?
  - When will the tests be performed?
- Preparing the tests may take months or years:
  - Collect requirements for test facilities
- Trace tests to requirements
  - Include tests in impact analysis

Principles of Requirements-Driven Testing

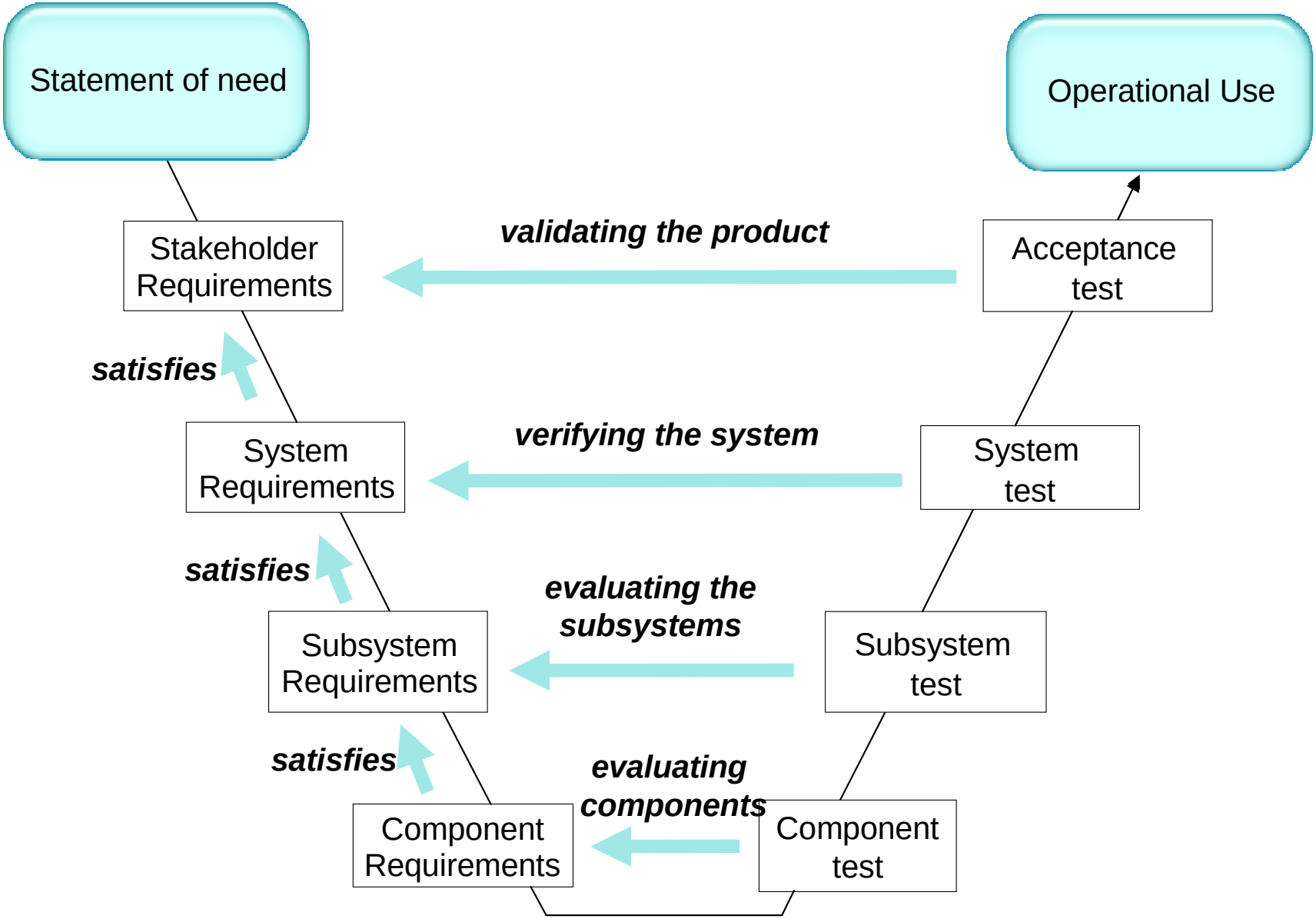
- Plan Tests Early
  - To understand the requirements better
- Conduct Tests Early
  - Phase injection vs. phase detection
- Relate Tests to Requirements
  - Assurance requirements are met
- Relate Defects to Requirements
  - Understand impact of defects
- Measure Progress against Requirements

Good Practice 5: Create, review and use traceability

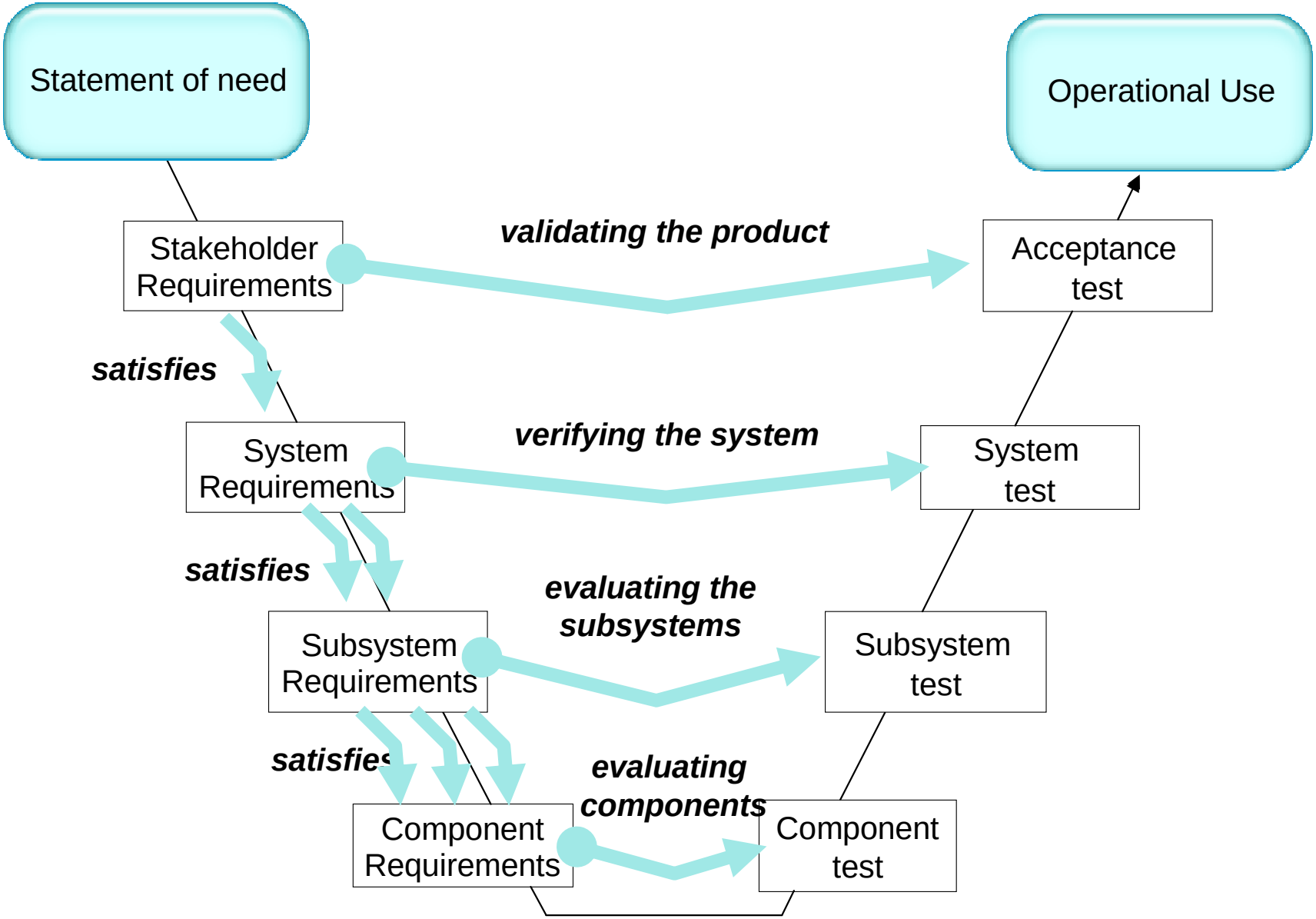
DEFINITION OF TRACEABILITY

- Documenting how high-level goals are transformed into low-level goals.
- Understanding how needs are satisfied
- Understand how requirements are qualified (tests, inspections, trials

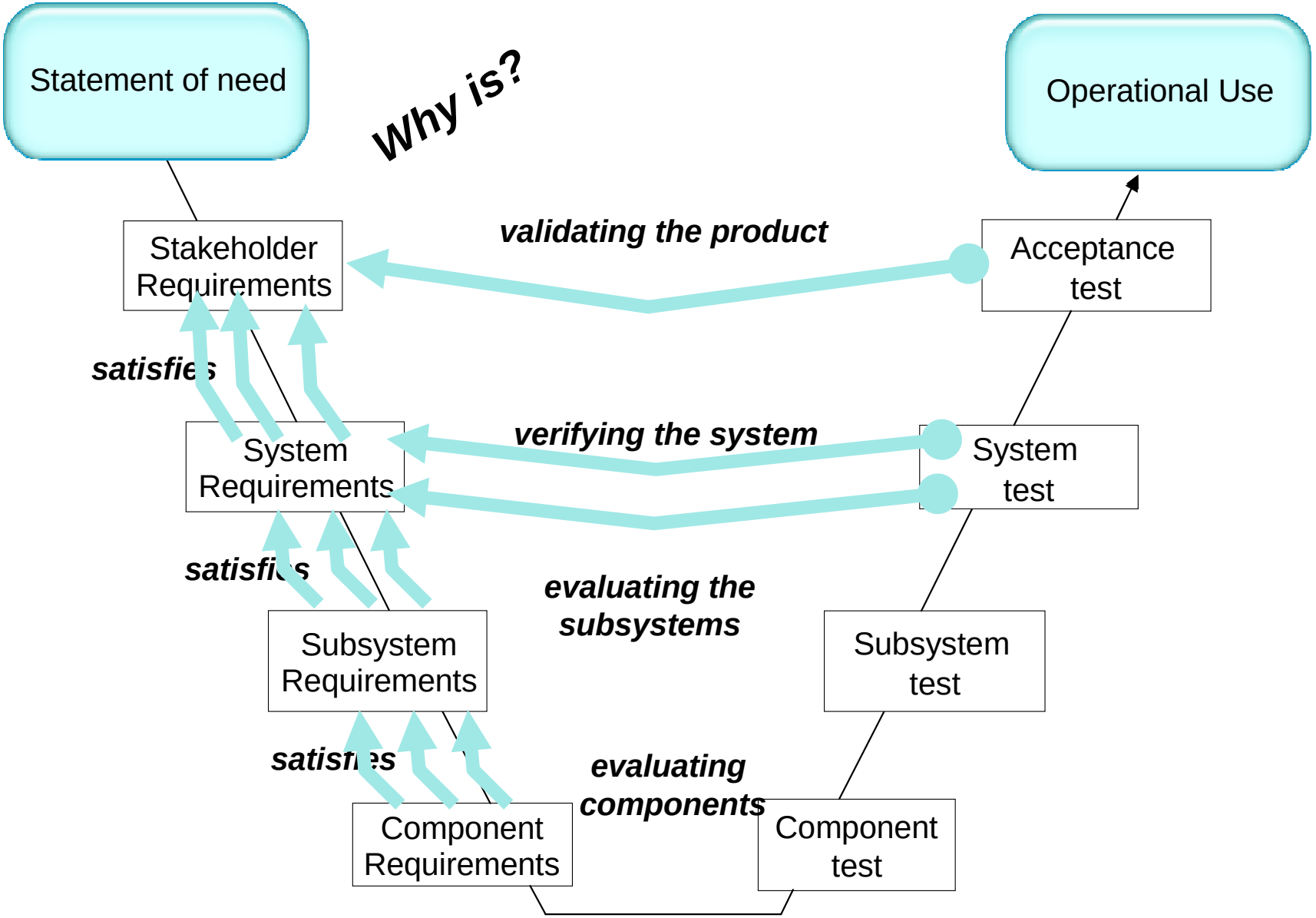
Traceability in the lifecycle



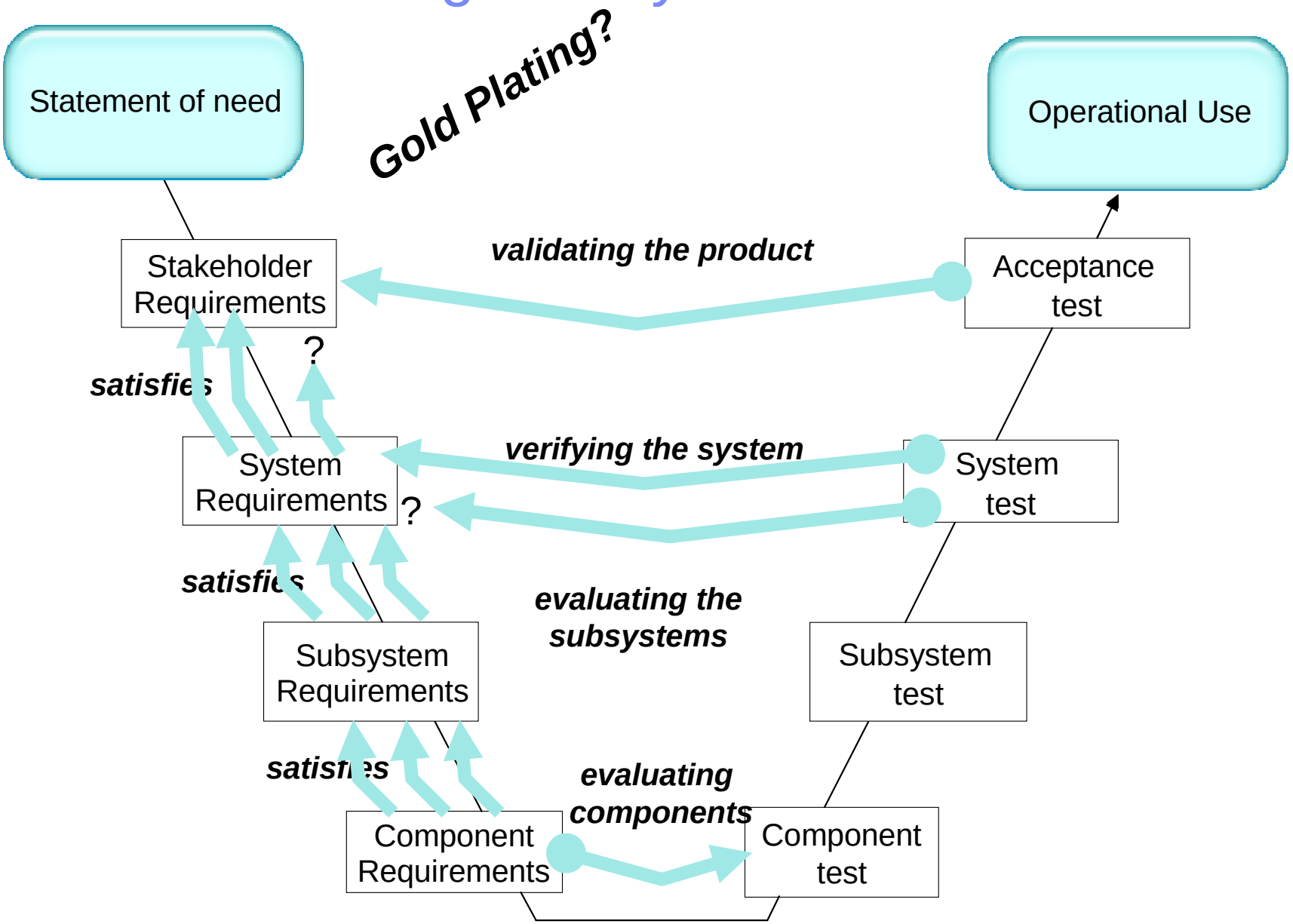
Impact Analysis



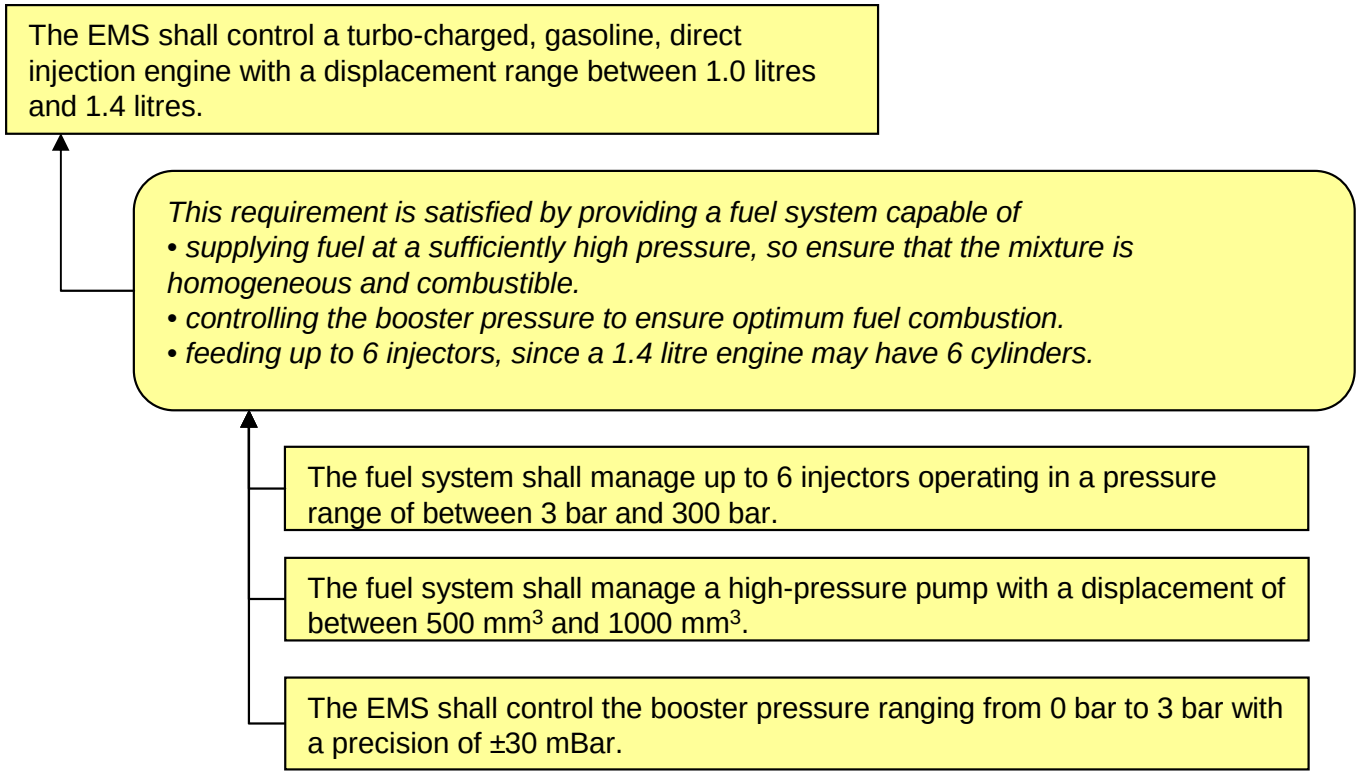
Derivation Analysis



Derivation Coverage Analysis



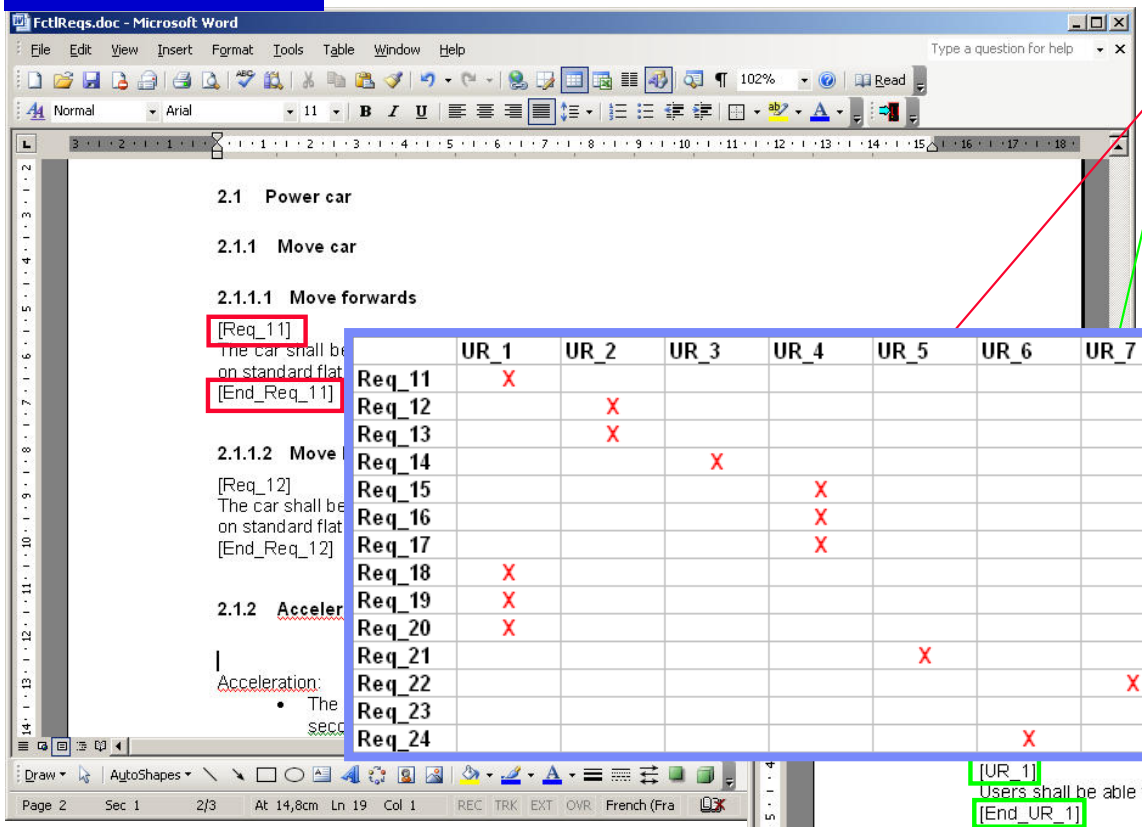
Three Criteria for Reviewing Traceability



1. ***Coverage:*** is every requirement traced?
2. ***Sufficiency***: are the traced lower-level requirements *sufficient* to satisfy the higher-level?
3. ***Necessity***: are all the traced lower-level requirements *necessary* to satisfy the higher-level?

Identify the element to trace

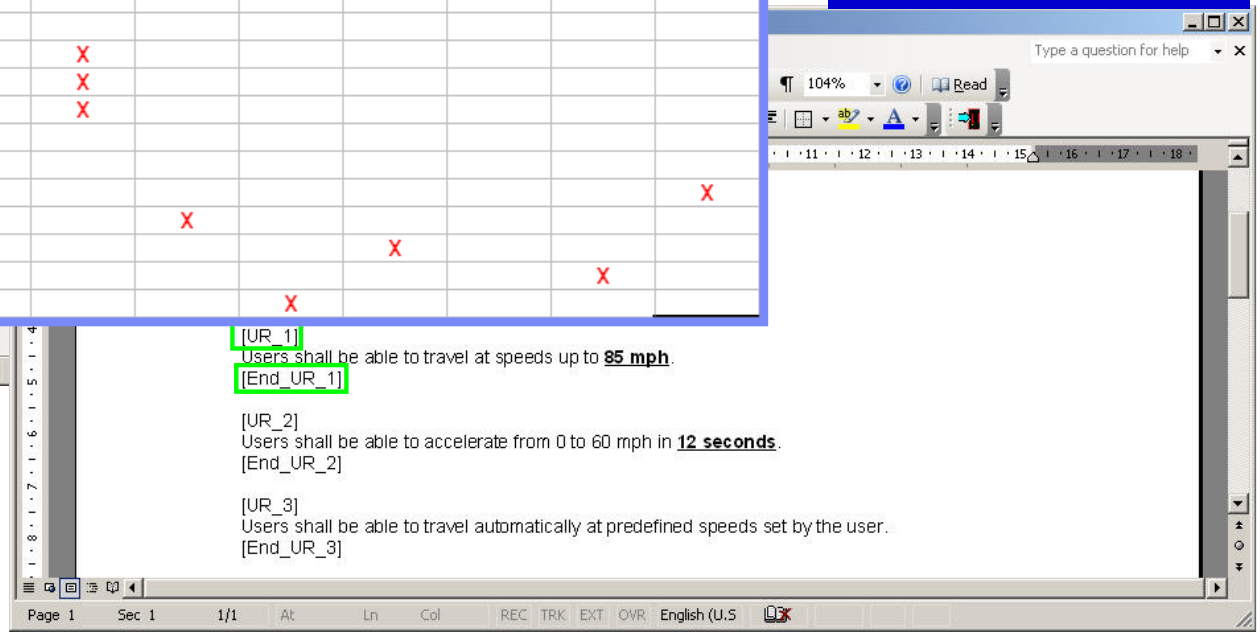
System Spec



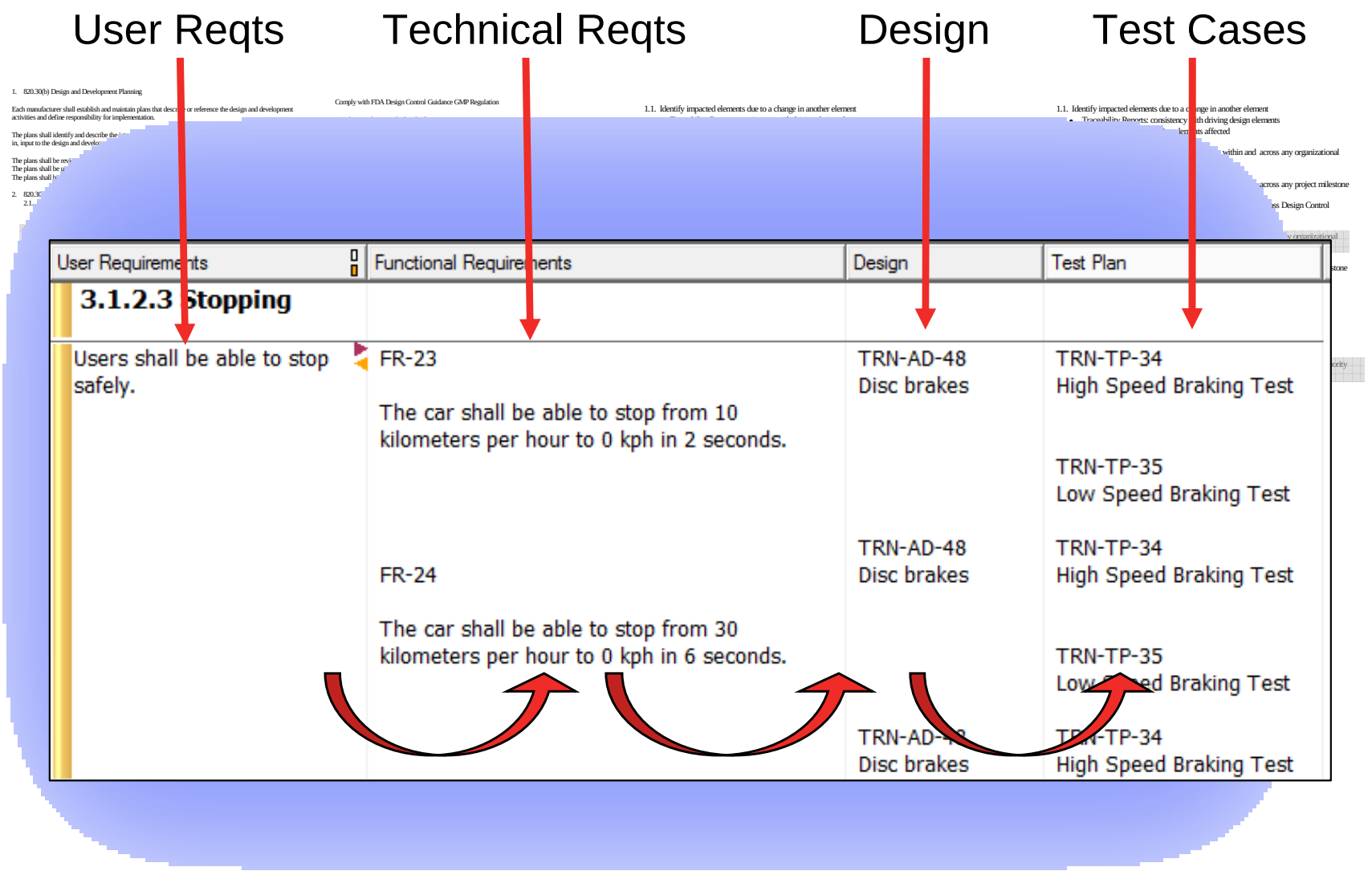
	UR_1	UR_2	UR_3	UR_4	UR_5	UR_6	UR_7	UR_8	UR_9	UR_10
Req_11	X									
Req_12		X								
Req_13		X								
Req_14			X							
Req_15				X						
Req_16				X						
Req_17				X						
Req_18	X									
Req_19	X									
Req_20	X									X
Req_21					X					
Req_22							X			
Req_23									X	
Req_24						X				

Need to use some tags to identify the element to trace.

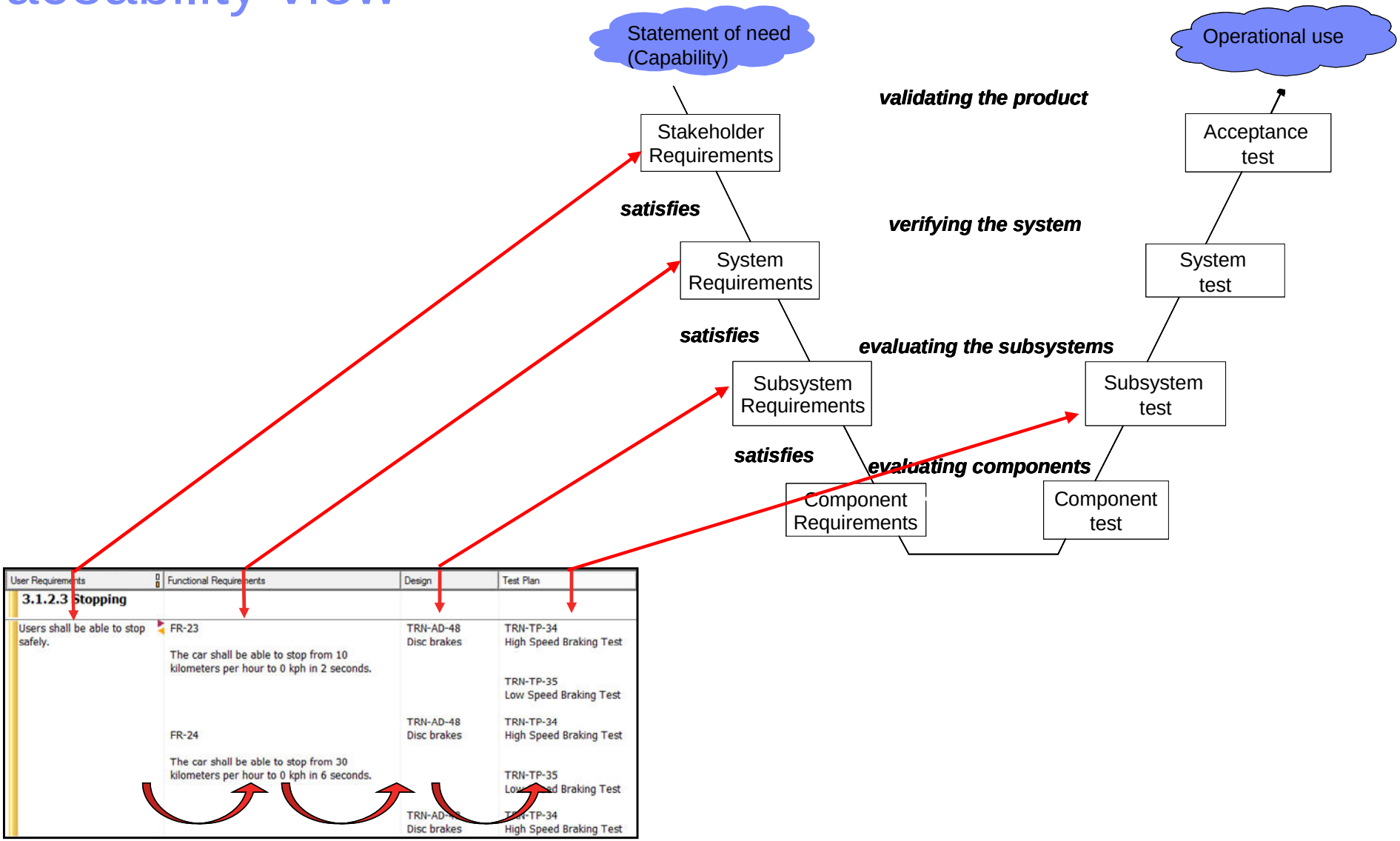
Customer's needs



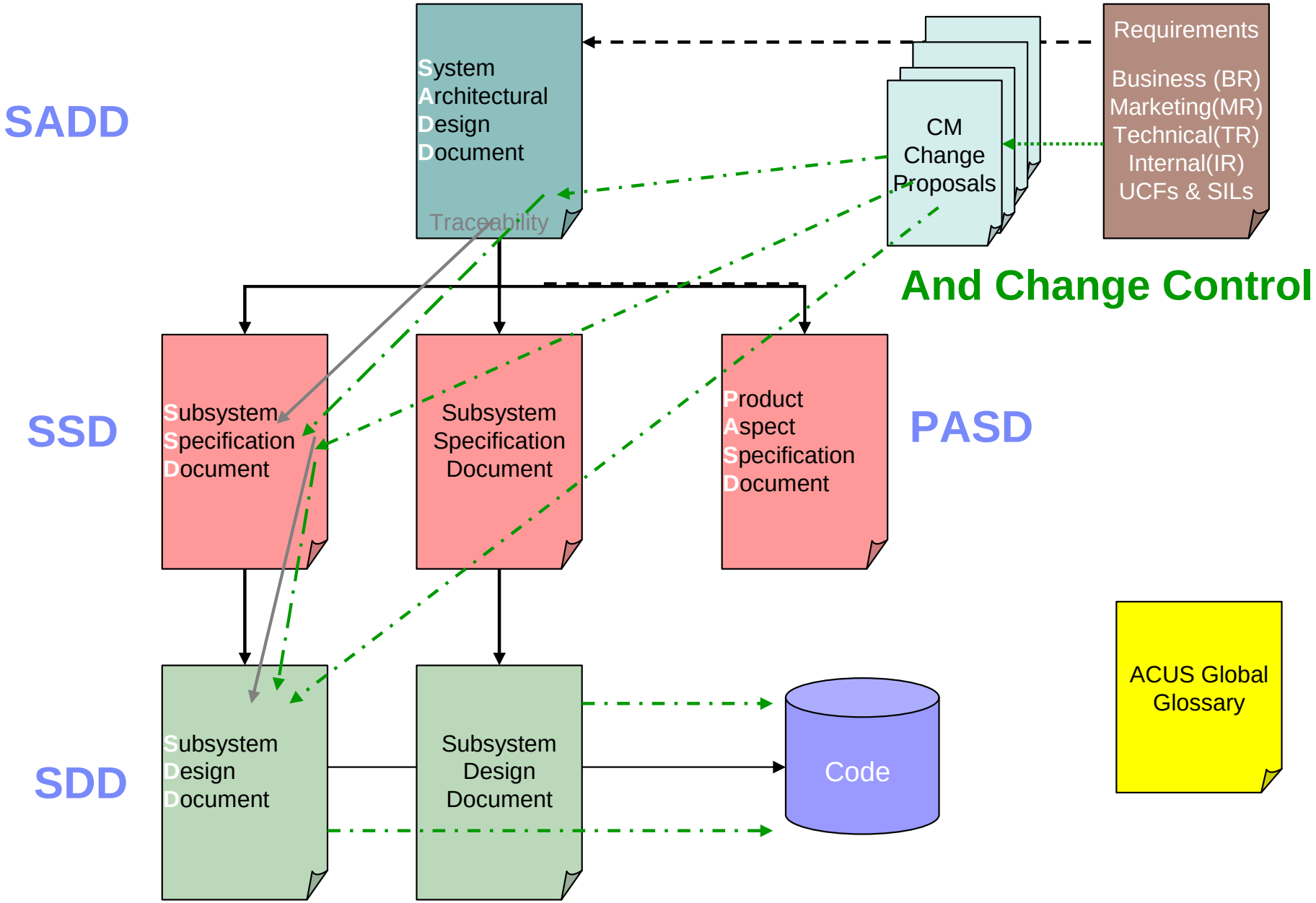
Traceability view



Traceability view



The Unisys Documentation Set

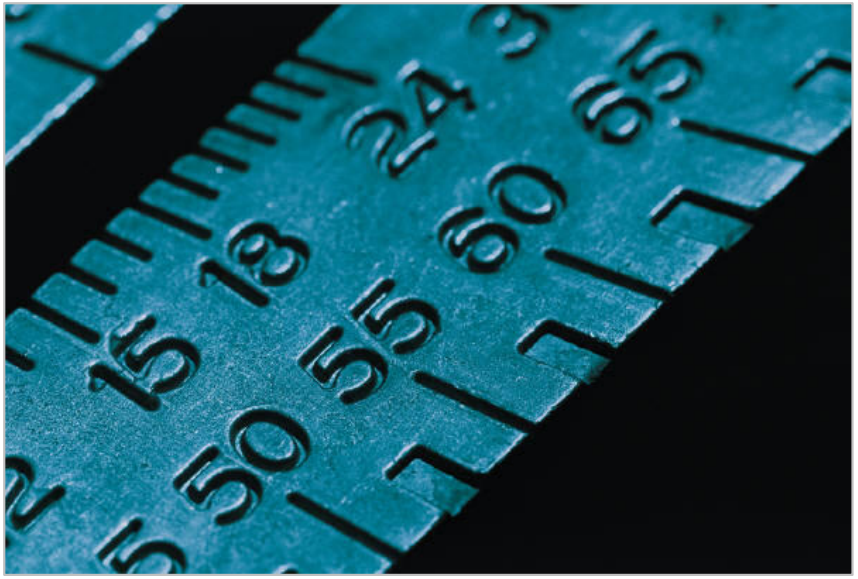


Questions

- **Can you quantify the improvement of your organisation's software development process over the past 5 years?**
- **Do you know if projects where you spend more relative effort in testing result in relatively fewer defects? How about projects where you spend more in requirements analysis?**
- **Based on what quantifiable information do you select development technologies for a project (modeling method, development environment, coding language, etc)?**

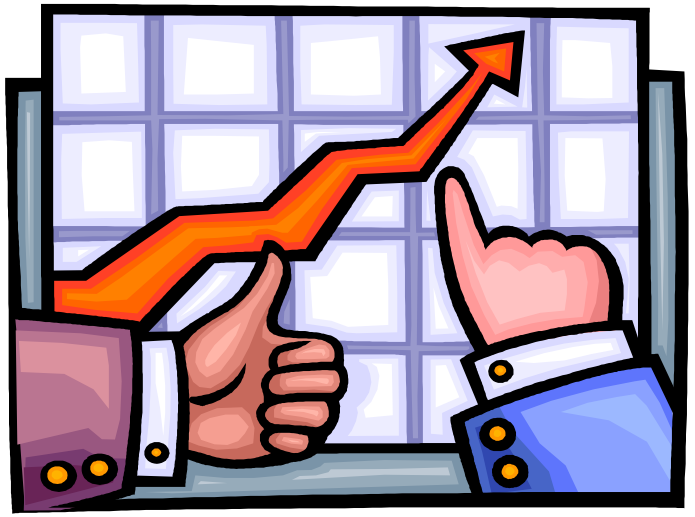
Measurement & Process Improvement

- You can't improve what you don't measure
- Keeping the metrics burden low
  - Automation of metrics
  - Quality
  - Productivity



Monitoring Progress based on requirements state

- Number (or %) of input requirements agreed
- Number (or %) of input requirements that have derived requirements linked to them
- Number (or %) of derived requirements in each requirement state (e.g. Draft, Proposed, Reviewed, Rejected)
- Number (or %) of derived requirements that have qualification activities linked to them
- Number (or %) of derived requirements in each qualification state (e.g. No qualification agreed, Qualification agreed, Qualification suspect)
- Number (or %) of input requirements with a change pending



Real Results

- Project
  - Methodology Process Mentor
  - Requirements Management in Rational DOORS
  - Build - some Agile techniques
  - Testing in HP Quality Centre
- Change Requests
  - Expected 20
  - Actual 3
- UAT
  - Expected 20
  - Actual 2
- Rework
  - Expected 20
  - Actual 0
- Requirements Acquittal
  - Expected 80%
  - Actual 125%
- Business Benefits Realisation
  - Expected 80%
  - Actual 110%

Graeme Higgins

Excuses!

- We’re too busy fighting fires to do requirements management
- We’re too busy delivering projects to do more requirements management
- We need to get on with the next project, there’s no time
- We don’t need another tool
- Our people are not skilled enough
- We’re doing the process/method first, then we’ll look at tool support
- …

Agenda

- The Case for Improved Requirements Management
- Good Requirements Management Practices
- No Excuses!